СОФИЙСКИ УНИВЕРСИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“

ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА

ДЪРЖАВЕН ИЗПИТ
ЗА ПОЛУЧАВАНЕ НА ОКС “БАКАЛАВЪР ПО КОМПЮТЪРНИ НАУКИ”

ЧАСТ I (ПРАКТИЧЕСКИ ЗАДАЧИ)
10.09.2018 г.

Моля, не пишете в тази таблица!			
Зад. 1		Зад. 5	
Зад. 2		Зад. 6	
Зад. 3		Зад. 7	
Зад. 4		Зад. 8	
Крайна оценка:			

Драги абсолвенти:

- Попълнете факултетния си номер в горния десен ъгъл на всички листа;
- Пишете само на предоставените листове без да ги разкопчавате;
- Ако имате нужда от допълнителен лист, можете да поискате от квесторите;
- Допълнителните листа трябва да се номерират, като номерата продължават тези от настоящия комплект;
- Всеки от допълнителните листа трябва да се надпише най-отгоре с вашия факултетен номер;
- **Решението на една задача трябва да бъде на същия лист, на който е и нейното условие (т.е. може да пишете отпред и отзад на листа със задачата, но не и на лист на друга задача);**
- Ако решението на задачата не се побира в един лист, трябва да поискате нов бял лист от квесторите. В такъв случай отново трябва да започнете своето решение на листа с условието на задачата и в края му да напишете „Продължава на лист № X”, където X е номерът на допълнителния лист, на който е вашето решение;
- Черновите трябва да бъдат маркирани, като най-отгоре на листа напишете „ЧЕРНОВА“;
- На един лист не може да има едновременно и чернова и белова;
- Времето за работа по изпита е 3 часа.

Изпитната комисия ви пожелава успешна работа!

Задача 1. *Задачата да се реши на един от езиците C или C++.*

Професор Х забравя постоянно паролата за своя таен сейф, която представлява последователност от цели числа. За да не е нужно да я помни, той скрива подсказки в кабинета си. На всеки ред от библиотеката му са подредени книги, най-много 20 реда книги, с до 30 книги на ред. Книгите на някои от редовете са подредени по азбучен ред на заглавията си, всяко от които е до 100 символа.

Паролата за сейфа на професор Х се определя от числата, които задават последователните дължини на думите в заглавията на книгите, разположени точно в средата на редовете, в които книгите са подредени в азбучен ред. Ако на реда има четен брой книги, за паролата се използва книгата, намираща се по-близо до началото на реда. Думите в заглавията на книгите са разделени от точно един интервал. Дължините на думите формират паролата в реда, в който се срещат, от най-горния към най-долния ред на библиотеката.
Библиотеката на Х може да се представи като двумерен масив **a** от низове с **m** реда по **n** низа всеки, представящи заглавията на книгите. Да се дефинира функция **`revealPassword`**, която по подадени **a**, **m** и **n**, извежда на стандартния изход паролата на професор Х като последователност от числа, разделени с по един интервал. Да се демонстрира употребата на функцията **`revealPassword`** в кратка програма.

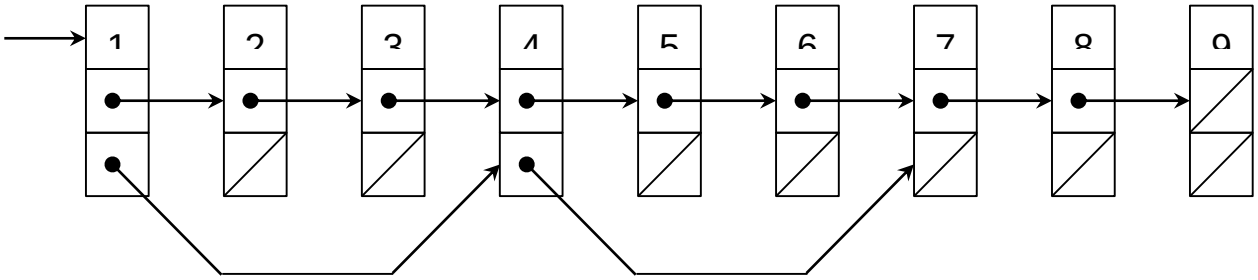
Пример:
Нека библиотеката на професор Х се състои от 3 реда с по 3 книги всеки както следва:

Алгебра	Аналитична геометрия	Математически анализ
Увод в програмирането	Обектно-ориентирано програмиране	Структури от данни и програмиране
Бази от данни	Изкуствен интелект	Функционално програмиране

Тогава паролата на професор Х е поредицата от числа 10 9 9 8, получена от дължините на подчертаните думи. Думите “Обектно-ориентирано програмиране” не участват в паролата, защото книгите на втория ред не са подредени по азбучен ред на заглавията си.

Задача 2. *Задачата да се реши на един от езиците C, C++ или Java. В началото на решението си посочете кой език сте избрали.*

Нека разгледаме следната структура от данни за бързо търсене, съхраняваща сортирана редица от цели числа:



Нека имаме списък с **n** елемента и нека с **k** означим горната цяла част на числото $\sqrt{n}$. Всеки възел на списъка съдържа цяло число и два указателя. Първият указател винаги сочи следващия елемент (ако има такъв). За възлите, намиращи се на индекс, кратен на *k*, вторият указател сочи възела, намиращ се на *k* позиции напред (ако има такъв). В горния пример **n** = **9**, **k** = **3**.

а) Да се напише функция **`readList`**, която по подаден път до текстов файл прочита от файла сортирана редица от цели числа и конструира списък от описания по-горе тип, съдържащ прочетените числа. Числата са записани във файла на един ред и са разделени с интервали.

б) Да се напише булева функция **`member`**, която по даден списък от описания тип и дадено цяло число проверява дали това число се съдържа в списъка. Функцията да реализира ефективен алгоритъм за търсене, който се възползва от особеностите на структурата, за да минимизира броя на обходените елементи.

Пример: ако в списъка от диаграмата по-горе търсим числото 6, при ефективния алгоритъм за търсене ще бъдат последователно обходени следните елементи: 1, 4, 7, 5, 6 и функцията **`member`** ще върне резултат “истина”.

*За реализацията на гореописаната структура от данни и за функциите **`readList`** и **`member`** не е позволено използване на библиотечни структури от данни и алгоритми, но е позволено използването на стандартните функции за работа с файлове.*

Задача 3. *Задачата да се реши на един от езиците Scheme или Haskell. По-долу оградете името на езика, който сте избрали за вашето решение.*

Даден е списък от списъци от числа **ll** и числова функция **f**. Числото **x** наричаме "корен" на **f**, ако **f(x) = 0**. Да се попълнят по подходящ начин празните полета по-долу, така че функцията **sumMaxRoots** да намира сумата на корените на **f** в този списък от **ll**, в който **f** има най-много корени. Ако има няколко такива списъка, функцията да връща сумата на корените в първия по ред списък. Ако функцията няма корен сред числата в списъците на **ll**, функцията да връща **0**.
Упътване: можете да използвате наготово функциите `apply`, `filter`, `foldr`, `length`, `map`, `max`, `maximum`, както и всички стандартни функции в R5RS за Scheme и Prelude за Haskell.

Scheme

```
(define (selectList l1 l2)
  (if ____________________________________________ l1 l2))

(define (sumMaxRoots f ll)
  (__________
      (__________ selectList __________
           (__________ (lambda (l)
                  (__________ (lambda (x) ____________________) l)) ll))))
```

Пример:
`(sumMaxRoots (lambda (x) (- (* x x x) x)) '((1 2 3) (-1 0 5) (1 4 -1))) → -1`

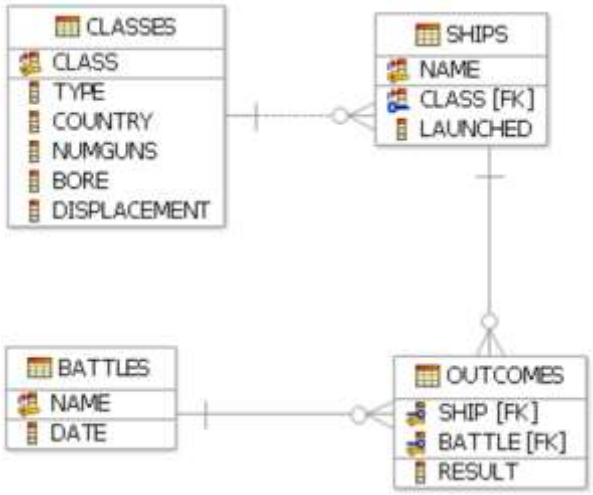
Haskell

```
selectList l1 l2 = if ____________________________________________ then l1 else l2

sumMaxRoots f ll =
  __________
      (__________ selectList __________
         (__________ (\l -> [ __________ | x <- l, ____________________ ]) ll))
```

Пример:
`sumMaxRoots (\x -> x^3 - x) [ [1, 2, 3], [-1, 0, 5], [1, 4, -1] ] → -1`

Задача 4. Дадена е базата от данни `Ships`, в която се съхранява информация за кораби и тяхното участие в битки по време на Втората световна война. Всеки кораб е построен по определен стереотип, определящ класа на кораба. Таблицата `Classes` съдържа информация за класовете кораби:



- `class` – име на клас, първичен ключ;
- `type` – тип (`'bb'` за бойни кораби, `'bc'` за бойни крайцери);
- `country` – държава, която строи такива кораби;
- `numguns` – брой оръдия, може да приема стойност `null`;
- `bore` – калибър на оръдието (в инчове), може да приема стойност `null`;
- `displacement` – водоизместимост (в тонове), може да приема стойност `null`.

Таблицата `Ships` съдържа информация за корабите:

- `name` – име на кораб, първичен ключ;
- `class` – име на клас, външен ключ към `Classes.class`;
- `launched` – година, в която корабът е пуснат на вода, може да приема стойност `null`.

Таблицата `Battles` съхранява информация за битките:

- `name` – име на битка, първичен ключ;
- `date` – дата на провеждане.

Таблицата `Outcomes` съдържа информация за резултата от участието на даден кораб в дадена битка. Атрибутите `ship` и `battle` заедно формират първичния ключ.

- `ship` – име на кораб, външен ключ към `Ships.name`;
- `battle` – име на битка, външен ключ към `Battles.name`;
- `result` – резултат (потънал – `'sunk'`, повреден – `'damaged'`, победил – `'ok'`).

Забележка за всички таблици: За всички атрибути, за които не е указано, че могат да приемат стойност `null`, да се счита, че съществува `not null` ограничение.

1. Да се напише заявка, която извежда име на клас, годината на първата битка, в която кораб на този клас е участвал, годината на последната битка, в която кораб на този клас е участвал, и броя на всички различни битки, в които кораби на този клас са участвали, само за тези класове, започващи с буквата `N`. Ако за даден клас няма кораб, който да е участвал в битка, за съответните години да се върне стойност `null`.

2. Да се напише заявка, която да изведе имената на тези битки, за които броят на корабите от тип `'bb'`, участвали в тази битка, е по-голям от броя на корабите от тип `'bc'`, участвали в същата битка. Битки, в които не е участвал нито един кораб, да не се извеждат в резултата.

Задача 5. Дадена е следната програма за ОС Linux, написана на езика C:

```
#include <unistd.h>
#include <string.h>
int main(void)
{
  char* buff = "Hello, world!\n";
  int p;
  if (fork() == 0) write(1, buff, strlen(buff));
  p = fork();
  write(1, buff, strlen(buff));
}
```

а) Колко пъти ще се изведе на стандартния изход текстът `"Hello, world!"` при изпълнението на програмата? Обосновете отговора си.

б) Нарисувайте кореновото дърво с върхове процесите, които ще се стартират в резултат от изпълнението на програмата и ребра двойките родител-наследник.

Задача 6. Даден е неориентиран свързан граф **G(V, E)** с тегловна функция **w : E → {a, b}**, където **a** и **b** са числа, такива че **0 < a < b**. Нека с **r** означим теглото на някое минимално покриващо дърво за графа **G**. Предложете линеен алгоритъм за намиране на числото **r**. Не е нужно алгоритъмът да намира минимално покриващо дърво. Дайте кратка обосновка на коректността и сложността по време на алгоритъма.

Упътване 1: Теглото на дървото се определя като сумата от теглата на ребрата му.

Упътване 2: Разгледайте подграфа **G'** на **G**, който съдържа само леките му ребра. Каква е зависимостта между броя на свързаните компоненти на **G'** и броя на тежките ребра в произволно минимално покриващо дърво на **G**?

Задача 7. Разглеждаме обикновени графи. *Хроматичното число* на даден граф $G = (V, E)$ е минималният брой цветове, с които може да се оцветят върховете му по такъв начин, че за всяко ребро (u,v), краищата му u и v са в различни цветове. Хроматичното число на G се бележи с $\chi(G)$. Казваме, че G е k-*хроматичен*, ако $\chi(G) = k$. Казваме, че G е *критично* k-*хроматичен*, ако $\chi(G) = k$, но за всеки връх v от G е вярно, че $\chi(G - v) = k - 1$. Тук "$G - v$" означава графа, получен от G чрез изтриването на връх v и на всички ребра, с които връх v е инцидентен. Докажете, че ако G е *критично* k-*хроматичен*, то за всеки връх v в G, степента на v е поне $k - 1$.

Упътване: Допуснете противното, тоест съставете твърдението-отрицание на това, което трябва да докажете. Сега изтрийте върха, за който става дума в твърдението-отрицание и използвайте факта, че G е *критично* k-*хроматичен*.

Задача 8. Пресметнете интеграла $\int_0^1 x^2 \operatorname{arctg} x \, dx$.

Чернова